(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105579967 B
(45)授权公告日 2019.09.03

(21)申请号 201480052983.1

(22)申请日 2014.09.10

(65)同一申请的已公布的文献号
申请公布号 CN 105579967 A

(43)申请公布日 2016.05.11

(30)优先权数据
14/043,562 2013.10.01 US

(85)PCT国际申请进入国家阶段日
2016.03.25

(86)PCT国际申请的申请数据
PCT/US2014/054966 2014.09.10

(87)PCT国际申请的公布数据
WO2015/050681 EN 2015.04.09

(73)专利权人 高通股份有限公司
地址 美国加利福尼亚州

(72)发明人 梅春惠
阿列克谢·弗拉狄米罗维奇·布尔德
陈林

(74)专利代理机构 北京律盟知识产权代理有限责任公司 11287
代理人 宋献涛

(51)Int.Cl.
G06F 9/52(2006.01)

(56)对比文件
CN 103207774 A,2013.07.17,
CN 1276890 A,2000.12.13,
CN 102640131 A,2012.08.15,
Minsoo Rhu等.CAPRI:Prediction of Compaction-Adequacy for Handling Control-Divergence in GPGPU Architectures.《International Symposium on Computer Architecture》.2012,61-71.
Xingxing Jin.Improving GPU SIMD Control Flow Efficiency via Hybrid Warp Size Mechanism.《http://ecommons.usask.ca/bitstream/handle/10388/ETD-2012-06-527/JIN-THESIS.pdf》.2012,1-82.
Wilson W.L.Fung等.Dynamic Warp Formation and Scheduling for Efficient GPU Control Flow.《40th IEEE/ACM International Symposium on Microarchitecture》.2007,407-416.

审查员 彭莉

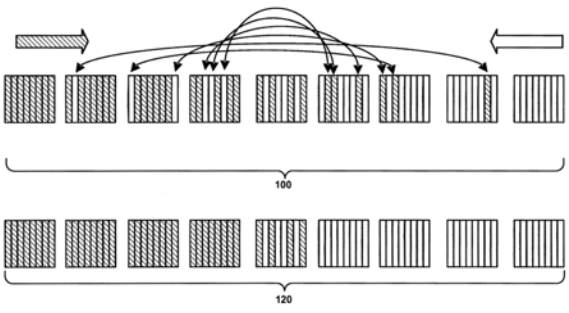
权利要求书5页 说明书15页 附图6页

(54)发明名称

GPU发散栅栏

(57)摘要

一种装置包含存储器和至少一个可编程处理器,所述至少一个可编程处理器经配置以为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真;暂停执行对于其所述表达式为真的具有对应线程的每一线程束;确定对于其所述表达式为真的所述多个线程束中的每一者的活动线程的数目;基于所述多个线程束中的每一者的活动线程的所述数目将对于其所述表达式为真的所述多个线程束排序;将所述多个线程束的第一线程束的活动线程的线程数据与所述多个线程束的第二线程束的非活动线程的线程数据调换;以及恢复执行对于其所述表达式为真的所述多个线程束中的所述至少一者。



CN 105579967 B

1.一种由至少一个可编程处理器执行的方法,所述方法包括:

为多个线程束中的每一线程束确定对于每一线程束的对应线程发散栅栏指令的布尔表达式自变量是否为真;

暂停执行具有对于其所述表达式为真的所述对应线程的每一线程束;

确定对于其所述表达式为真的所述多个线程束中的每一者的活动线程的数目,其中所述活动线程已采用分支;

基于所述多个线程束中的每一者中的活动线程的所述数目将对于其所述表达式为真的所述多个线程束排序,以产生多个经排序的线程束;

基于所述多个经排序的线程束中的每一者中的活动线程的所述数目将所述多个线程束的具有更多非活动线程的第一线程束的活动线程的线程数据与所述多个线程束的具有更多活动线程的第二线程束的非活动线程的线程数据调换;以及

恢复执行具有对于其所述表达式为真的多个线程中的一者的所述多个经排序的线程束中的至少一者。

2.根据权利要求1所述的方法,所述方法进一步包括:

在恢复执行所述多个经排序的线程束中的所述至少一者之前,调换对于其所述表达式为真的所述多个线程的每线程上下文数据。

3.根据权利要求1所述的方法,其中所述至少一个可编程处理器包括图形处理单元GPU。

4.根据权利要求1所述的方法,其中所述活动线程的所述线程数据包括所述活动线程的寄存器数据,且

其中所述非活动线程的所述线程数据包括所述非活动线程的寄存器数据。

5.根据权利要求1所述的方法,其中将所述多个线程束排序包括使用插入排序将所述多个线程束排序。

6.根据权利要求1所述的方法,其进一步包括:

为对于其所述表达式为真的所述多个线程束中的每一线程束确定多个发散栅栏中的相关联发散栅栏;

基于每一线程束的相关联发散栅栏将所述多个线程束中的每一线程束分组到多个压缩池中,

其中将所述多个线程束排序包括将属于所述多个压缩池中的同一者的所述多个线程束排序,

其中所述第一线程束以及所述第二线程束包括属于所述多个压缩池中的所述同一者的线程束,且

其中恢复执行对于其所述表达式为真的所述多个线程束中的所述至少一者包括恢复执行同一个压缩池的至少一个线程束。

7.根据权利要求6所述的方法,其进一步包括:

基于与所述多个线程束中的每一者相关联的所述发散栅栏将前缀指派到所述多个线程束中的每一者,

其中将所述多个线程束分组到所述至少一个压缩池中包括基于所指派的所述前缀将所述多个线程束分组到至少一个压缩池中。

8.根据权利要求1所述的方法，其中将活动线程的所述线程数据与非活动线程的所述线程数据调换继续进行，直到非活动线程束不可形成为止。

9.根据权利要求1所述的方法，其进一步包括：

确定所述多个线程束包含具有全部活动线程的线程束；以及恢复执行具有全部活动线程的所述线程束。

10.根据权利要求1所述的方法，其中将对于其所述表达式为真的所述多个线程束排序包括：

将所述多个线程束存储在队列中；

基于活动线程的所述数目将对于其所述表达式为真的所述队列中的所述多个线程束排序；以及

将所述多个经排序的线程束存储在所述队列中。

11.根据权利要求1所述的方法，其进一步包括：

确定其中很可能发生发散的位置和执行于多个线程束上的核心内的将显著影响性能的位置中的至少一者；以及

将发散栅栏指令插入到所述核心中的所述至少一个位置处，

其中所述布尔表达式与所述发散栅栏指令相关联。

12.一种用于减少执行在图形处理单元上的线程的发散的设备，其包括：

用于为多个线程束中的每一线程束确定对于每一线程束的对应线程发散栅栏指令的布尔表达式自变量是否为真的装置；

用于暂停执行具有对于其所述表达式为真的所述对应线程的每一线程束的装置；

用于确定对于其所述表达式为真的所述多个线程束中的每一者的活动线程的数目的装置，其中所述活动线程已采用分支；

用于基于所述多个线程束中的每一者中的活动线程的所述数目将对于其所述表达式为真的所述多个线程束排序，以产生多个经排序的线程束的装置；

用于基于所述多个经排序的线程束中的每一者中的活动线程的所述数目将所述多个线程束的具有更多非活动线程的第一线程束的活动线程的线程数据与所述多个线程束的具有更多活动线程的第二线程束的非活动线程的线程数据调换的装置；以及

用于恢复执行具有对于其所述表达式为真的多个线程中的一者的所述多个经排序的线程束中的至少一者的装置。

13.根据权利要求12所述的设备，所述设备进一步包括：

用于在恢复执行所述多个经排序的线程束中的所述至少一者之前，调换对于其所述表达式为真的所述多个线程的每线程上下文数据的装置。

14.根据权利要求12所述的设备，其中所述设备包括图形处理单元GPU。

15.根据权利要求12所述的设备，其中所述活动线程的所述线程数据包括所述活动线程的寄存器数据，且

其中所述非活动线程的所述线程数据包括所述非活动线程的寄存器数据。

16.根据权利要求12所述的设备，其中用于将所述多个线程束排序的所述装置包括用于使用插入排序将所述多个线程束排序的装置。

17.根据权利要求12所述的设备，其进一步包括：

用于为对于其所述表达式为真的所述多个线程束中的每一线程束确定多个发散栅栏中的相关联发散栅栏的装置；

用于基于每一线程束的相关联发散栅栏将所述多个线程束中的每一线程束分组到多个压缩池中的装置，

其中用于将所述多个线程束排序的所述装置包括用于将属于所述多个压缩池中的同一者的所述多个线程束排序的装置，

其中所述第一线程束以及所述第二线程束包括属于所述多个压缩池中的所述同一者的线程束，且

其中用于恢复执行对于其所述表达式为真的所述多个线程束中的所述至少一者的所述装置包括用于恢复执行所述同一个压缩池的至少一个线程束的装置。

18.根据权利要求17所述的设备，其进一步包括：

用于基于与所述多个线程束中的每一者相关联的所述发散栅栏将前缀指派到所述多个线程束中的每一者的装置，

其中用于将所述多个线程束分组到所述至少一个压缩池中的所述装置包括用于基于所指派之所述前缀将所述多个线程束分组到至少一个压缩池中的装置。

19.根据权利要求12所述的设备，其中用于将活动线程的所述线程数据与非活动线程的所述线程数据调换继续进行，直到非活动线程束不可形成为止的所述装置。

20.根据权利要求12所述的设备，其进一步包括：

用于确定所述多个线程束包含具有全部活动线程的线程束的装置；以及

用于恢复执行具有全部活动线程的所述线程束的装置。

21.根据权利要求12所述的设备，其中用于将对于其所述表达式为真的所述多个线程束排序的所述装置包括：

用于将所述多个线程束存储在队列中的装置；

用于基于活动线程的所述数目将对于其所述表达式为真的所述队列中的所述多个线程束排序的装置；以及

用于将所述多个经排序的线程束存储在所述队列中的装置。

22.根据权利要求12所述的设备，其进一步包括：

用于确定其中很可能发生发散的位置和执行于多个线程束上的核心内的将显著影响性能的位置中的至少一者的装置；以及

用于将发散栅栏指令插入到所述核心中的所述至少一个位置处的装置，

其中布尔表达式与所述发散栅栏指令相关联。

23.一种非暂时性电脑可读储存媒体，其包含指令，当所述指令执行时，使至少一个可编程处理器进行以下操作：

为多个线程束中的每一线程束确定对于每一线程束的对应线程发散栅栏指令的布尔表达式自变量是否为真；

暂停执行具有对于其所述表达式为真的所述对应线程的每一线程束，其中活动线程已采用分支；

确定对于其所述表达式为真的所述多个线程束中的每一者的活动线程的数目；

基于所述多个线程束中的每一者中的活动线程的所述数目将对于其所述表达式为真

的所述多个线程束排序,以产生多个经排序的线程束;

基于所述多个经排序的线程束中的每一者中的活动线程的所述数目将所述多个线程束的具有更多非活动线程的第一线程束的活动线程的线程数据与所述多个线程束的具有更多活动线程的第二线程束的非活动线程的线程数据调换;以及

恢复执行具有对于其所述表达式为真的多个线程中的一者的所述多个经排序的线程束中的至少一者。

24.一种用于减少执行在图形处理单元上的线程的发散的设备,其包括:

存储器;以及

至少一个可编程处理器,其经配置以进行以下操作:

为多个线程束中的每一线程束确定对于每一线程束的对应线程发散栅栏指令的布尔表达式自变量是否为真;

暂停执行具有对于其所述表达式为真的所述对应线程的每一线程束;

确定对于其所述表达式为真的所述多个线程束中的每一者的活动线程的数目,其中所述活动线程已采用分支;

基于所述多个线程束中的每一者中的活动线程的所述数目将对于其所述表达式为真的所述多个线程束排序,以产生多个经排序的线程束;

基于所述多个经排序的线程束中的每一者中的活动线程的所述数目将所述多个线程束的具有更多非活动线程的第一线程束的活动线程的线程数据与所述多个线程束的具有更多活动线程的第二线程束的非活动线程的线程数据调换;以及

恢复执行具有对于其所述表达式为真的多个线程中的一者的所述多个经排序的线程束中的至少一者。

25.根据权利要求24所述的设备,其中所述至少一个可编程处理器进一步继续进行以下操作:

在恢复执行所述经排序的多个线程束中的所述至少一者之前,调换对于其所述表达式为真的所述多个线程的每线程上下文数据。

26.根据权利要求24所述的设备,其中所述设备包括图形处理单元GPU。

27.根据权利要求24所述的设备,其中所述活动线程的所述线程数据包括所述活动线程的寄存器数据,且

其中所述非活动线程的所述线程数据包括所述非活动线程的寄存器数据。

28.根据权利要求24所述的设备,其中所述至少一个可编程处理器经进一步配置以进行以下操作:

为对于其所述表达式为真的所述多个线程束中的每一线程束确定多个发散栅栏中的相关联发散栅栏;

基于每一线程束的相关联发散栅栏将所述多个线程束中的每一线程束分组到多个压缩池中,

其中为将所述多个线程束排序,所述至少一个可编程处理器经进一步配置以排序所述多个线程束包括将属于所述多个压缩池中的同一者的所述多个线程束排序,

其中所述第一线程束以及所述第二线程束包括属于所述多个压缩池中的所述同一者的线程束,且

其中为恢复执行对于其所述表达式为真的所述多个线程束中的所述至少一者，所述至少一个可编程处理器经配置以恢复执行所述同一个压缩池的至少一个线程束。

29.根据权利要求28所述的设备，其中所述至少一个可编程处理器经进一步配置以进行以下操作：

基于与所述多个线程束中的每一者相关联的所述发散栅栏将前缀指派到所述多个线程束中的每一者，

其中使所述至少一个可编程处理器将所述多个线程束分组到所述至少一个压缩池中的所述指令包括基于所指派的所述前缀使所述至少一个可编程处理器将所述多个线程束分组到至少一个压缩池中的指令。

30.根据权利要求24所述的设备，其中为将对于其所述表达式为真的所述多个线程束排序，所述至少一个可编程处理器经配置以进行以下操作：

将所述多个线程束存储在队列中；

基于活动线程的所述数目将对于其所述表达式为真的所述队列中的所述多个线程束排序；以及

将所述多个经排序的线程束存储在所述队列中。

GPU发散栅栏

技术领域

[0001] 本发明涉及图形处理，且更确切地说，涉及用于管理执行图形处理单元(GPU)上的线程的技术。

背景技术

[0002] 最近，已存在一种朝向所谓的通用GPU(GPGPU)的转变。不同于执行图形渲染的传统GPU，GPGPU可经配置以执行通常被称作“核心”的通用任务或程序。一些类型的任务可更好地适合于特定类型的处理器，例如中央处理单元(CPU)或GPU。CPU可更好地适合于具有更多分支、跳转和条件性逻辑的任务，而GPU可适合于高度平行任务及/或具有多个浮点计算的任务。由于多个GPU具有SIMD硬件架构，因此GPU还可包含执行SIMD(单指令多数据)指令的能力。当GPU执行SIMD指令时，GPU可对多个数据值执行由指令指示的同一操作。通常，GPU具有多个执行单元，所述多个执行单元能够并行执行由SIMD指令指示的操作。

发明内容

[0003] 本发明的技术提供用于降低执行于图形处理单元(GPU)上的线程当中的发散的技术。GPU可包括支持被称作“发散栅栏”指令的指令。发散栅栏指令尝试将来自多个线程束的发散线程分组到新线程束中，使得线程执行同一指令，进而改善GPU性能。

[0004] 在一个实例中，本发明描述一种方法，所述方法包括：为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真；暂停执行具有对于其表达式为真的对应线程的每一线程束；确定对于其表达式为真的多个线程束中的每一者的活动线程的数目；基于多个线程束中的每一者的活动线程的数目将对于其表达式为真的多个线程束排序；将多个线程束的第一线程束的活动线程的线程数据与多个线程束的第二线程束的非活动线程的线程数据调换；以及恢复执行对于其表达式为真的多个线程束中的至少一者。

[0005] 在另一实例中，本发明描述一种包含存储器和至少一个可编程处理器的装置，所述至少一个可编程处理器经配置以：为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真；暂停执行具有对于其表达式为真的对应线程的每一线程束；确定对于其表达式为真的多个线程束中的每一者的活动线程的数目；基于多个线程束中的每一者中的活动线程的数目将对于其表达式为真的多个线程束排序；将多个线程束的第一线程束的活动线程的线程数据与多个线程束的第二线程束的非活动线程的线程数据调换；以及恢复执行对于其表达式为真的多个线程束中的至少一者。

[0006] 在另一实例中，本发明描述一种设备，所述设备包含：用于为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真的装置；用于暂停执行具有对于其表达式为真的对应线程的每一线程束的装置；用于确定对于其表达式为真的多个线程束中的每一者的活动线程的数目的装置；用于基于多个线程束中的每一者中的活动线程的数目将对于其表达式为真的多个线程束排序的装置；用于将多个线程束的第一线程束的活动线程的线程数据与多个线程束的第二线程束的非活动线程的线程数据调换的装置；用于

恢复执行对于其表达式为真的多个线程束中的至少一者的装置。

[0007] 在另一实例中，本发明描述一种存储指令的非暂时性电脑可读储存媒体，当所述指令执行时，使至少一个可编程处理器进行以下操作：为对于其表达式为真的多个线程束中的每一线程束确定多个发散栅栏中的相关联发散栅栏；基于每一线程束的相关联发散栅栏将多个线程束中的每一线程束分组到多个压缩池中，其中使至少一处理器将多个线程束排序的指令包括至少一处理器将属于多个压缩池中的同一者的多个线程束排序的指令，其中第一线程束和第二线程束属于多个压缩池中的同一者，且其中使至少一处理器恢复执行对于其条件为真的多个线程束中的至少一者包括恢复执行同一个压缩池的至少一个线程束。

[0008] 在附图及以下描述中阐述本发明的一或多个实例的细节。本发明的其它特征、目标和优点将从所述描述和图式以及权利要求书而显而易见。

附图说明

[0009] 图1为说明根据本发明的技术的可支持GPU发散栅栏指令的执行的实例计算装置的框图。

[0010] 图2为说明根据本发明的技术的在多个处理元件上执行的线程束的框图。

[0011] 图3为说明根据本发明的技术的基于每一线程束内的活动线程的数目将线程束排序的概念图。

[0012] 图4为说明用于将来自一个线程束的活动线程与来自另一线程束的非活动线程调换的技术的概念图。

[0013] 图5为说明根据本发明的技术的用于处置多个发散栅栏指令的技术的概念图。

[0014] 图6为说明根据本发明的技术的用于执行发散栅栏指令的技术的流程图。

具体实施方式

[0015] 本发明涉及用于减少执行在图形处理单元(GPU)上的线程的发散的技术。GPU可包括被称作处理元件(PE)的多个执行单元。被称作“核心”的程序可在GPU的一或多个PE上执行。应用程序可将核心划分到多个线程中，所述线程构成GPU的工作的基础单元。GPU调度器可进一步将线程一起分组到被称作“线程束”的线程群组中。线程束可包含一些图形架构中的某一数目的线程(例如，32个线程)。

[0016] GPU的驱动程序或调度器产生线程以执行GPU上的核心。线程为GPU上待处理的数据的基础单元，且不应与CPU线程混淆。调度器可将每一线程指派到GPU的执行单元。执行单元(也被称作处理元件(“PE”)或着色器)为能够对多个数据值并行执行同一指令的SIMD单元。

[0017] 一般来说，线程束的每一线程执行同一指令。程序计数器(PC)存储每一线程将执行的指令的存储器地址。一般来说，对于线程束的线程中的每一者可存在单一PC。对于每一线程束具有单一PC允许线程中的每一者同时执行，只要线程中的每一者不需要执行不同指令。

[0018] 多个GPU现在包括用以执行流控制指令(例如，用以执行分支、跳转、转至)和其它流控制指令的能力。流控制指令可以多种方式更改程序执行的流。在不具有流控制指令的

程序或核心中，PE可从头到尾地执行核心的指令。在PE结束执行指令之后，GPU将PC的值设定为存储器中的下一个指令的地址(通常通过逐一增加PC值)，且PE执行下一个指令。执行程序的过程以此方式继续执行不具有流控制指令的程序，直到程序到达退出点为止，在所述点处执行终止。

[0019] 执行流控制指令可使PE执行除递增PC值以外的地址处的后续指令。而非执行递增PC值的地址处的后续指令，执行流控制指令的PE可执行具有不同PC地址(例如子例程的地址等)的后续指令。因此，据称流控制指令更改程序的执行“流”。

[0020] 流控制指令的实例包括子例程调用、分支、返回、跳跃等。在不同实例中，PE跳转到的指令地址(即经指派至PC的地址)可基于运行时处在线程之间变化的数据的值。流控制指令还可与每一PE分别评估的布尔表达式相关联。布尔表达式为产生评估真或假的布尔值的表达式。布尔表达式可包含布尔运算符，例如“且”、“或”、“非”、“异或(XOR)”等。布尔表达式还可包含运算检验，例如大于、小于、等于、不等于、大于或等于、小于或等于等。布尔表达式的真值或假值可取决于随一个线程到另一线程而不同的数据或值。

[0021] 因此，有可能一个线程跳转到，且执行不同于同一线程束内的另一线程的指令。然而，如上所述，对于线程序仅存在一个PC。其中线程束的两个或两个以上线程执行不同指令的条件被称作“发散”。当发散出现时，线程的一些组可持续执行同一指令。然而，还可存在执行不同指令的多组线程。

[0022] 作为线程发散的实例，线程束的第一线程和第二线程可执行流控制指令，例如“if-else”语句或循环语句。第一线程执行的后续指令可基于存储于第一线程的寄存器中的数据的值。类似地，第二线程的后续指令可基于存储于第二线程的寄存器中的数据的值。如果第一和第二线程具有不同寄存器数据，那么第一和第二线程可跳转到与不同指令地址相关联的不同后续指令。

[0023] 在线程束线程发散的情况下，线程可采用控制流块的不同分支，例如“if-else”语句。在循环语句的情况下，线程束线程还可在不同时间(例如，在执行不同数目的循环的迭代之后)退出循环语句。

[0024] 当线程束线程(例如)由于采用if-else语句的不同分支或执行不同数目的循环的迭代而变得发散时，GPU使由发散导致的不同执行路径中的每一者序列化。也就是说，GPU确定“活动”的线程且正在执行同一指令。活动线程继续在与每一线程相关联的PE上执行，直到线程结束执行或到达栅栏(例如下文将极详细地的论述的发散栅栏指令)为止。

[0025] 在序列化期间，GPU还确定当前不执行的线程，且将那些非活动线程及其相关联的PE设定为空闲。当PE设定为空闲时，非活动线程不执行，此损害GPU性能。在一些情况下，发散线程可进一步发散，即可存在多个“层级”或“嵌套式发散”。为处置嵌套式发散，GPU使用收敛堆栈来追踪嵌套式分支和循环。GPU首先处置最深或最内层的发散，且执行具有最深层级的发散，直到执行结束或暂停为止。GPU随后将所述层级的发散从收敛堆栈移除，且重复执行收敛堆栈上的最内部的其余线程的过程，且将结束的线程从收敛堆栈移除。一旦线程结束执行分支或循环，GPU可将线程重组或收敛回在一起，以形成不再发散的线程束。

[0026] 本发明的技术介绍GPU可支持的被称作“发散栅栏”的指令。在不同实例中，应用软件编程接口(API)可包含支持发散栅栏指令。此类API可包含开放计算语言(OpenCL)、开放图形语言(OpenGL)和微软DirectX API。当编程具有特定API的GPU时，编程器可在发散很可

能显著影响性能的代码点处插入使GPU执行发散栅栏指令的发散栅栏函数调用。GPU驱动程序或编译程序还可自动检测发散很可能显著影响性能的代码点，且可在那些代码点处插入发散栅栏指令。

[0027] CPU随后将包含发散栅栏指令的核心的代码发射到GPU以供执行。GPU随后执行核心代码，直到其遇到发散栅栏指令为止。每一发散栅栏指令使GPU评估布尔表达式。如果GPU将布尔表达式评估为真，那么GPU暂停执行线程束。GPU切换到且开始执行另一线程束。GPU继续进行执行线程束的过程，直到核心的全部线程束结束执行或暂停(例如，归因于执行发散栅栏指令)为止。一旦全部线程束结束执行或暂停，GPU尝试消除由于执行发散栅栏指令而当前暂停的线程束当中的发散。

[0028] 当GPU执行发散栅栏指令且暂停执行线程束时，GPU将线程束插入到由于已执行发散栅栏指令而当前暂停的线程束的队列中。一经放置到队列中，GPU基于每一线程束中的活动线程的数目使用插入排序将队列中的线程束排序，且使用插入排序将队列中的暂停的线程束中的每一者排序。当全部线程束暂停且以队列形式排序(或结束)之后，GPU随后尝试消除执行核心的线程束的线程当中的发散。消除线程束的线程当中的发散的过程被称作“线程压缩”。

[0029] 在线程压缩期间，GPU尝试通过将具有更多非活动线程的当前活动线程束与来自具有更多活动线程的线程束的非活动线程调换来形成具有没有发散或更低发散的线程的线程束。当调换来自不同线程束的线程时，GPU使用线程束排序队列以将更换的数据量降到最小。在导致新线程束的形成的GPU线程压缩期间，一旦具有全部活动线程的线程束形成，GPU可继续执行每一新线程束。以此方式，经配置以支持发散栅栏指令的GPU可降低线程束线程发散且改善GPU性能。

[0030] 图1为说明根据本发明的技术的可支持GPU发散栅栏指令的执行的实例计算装置的框图。图1包含计算装置2。计算装置2可包括个人计算机、桌上型计算机、膝上型计算机、计算机工作站、平板计算装置、视频游戏平台或控制台、无线通信装置(例如，移动电话、蜂窝式电话、卫星电话和/或移动电话手持机)、陆线电话、因特网电话、手持式装置(例如，便携式视频游戏装置或个人数字助理(PDA))、个人音乐播放器、视频播放器、显示装置、电视、电视机顶盒、服务器、中间网络装置、主机计算机或处理及/或显示图形数据的任何其它类型的装置。

[0031] 如图1的实例中所说明，计算装置2包含CPU 16、系统存储器14、图形处理单元(GPU)12以及编译程序/驱动程序18。CPU 16可执行各种类型的应用程序。应用程序的实例包含网络浏览器、电子邮件应用程序、电子数据表、视频游戏或产生可视对象以供显示的其它应用程序。用于执行一或多个应用程序的指令可存储在系统存储器14内。

[0032] CPU 16还可执行编译程序/驱动程序18。编译程序/驱动程序18可包括控制GPU 12的相互作用的编译程序和/或驱动程序。编译程序/驱动程序18可采用程序代码(例如写入特定图形应用程序编程接口(API)中的代码)，且将代码转译到核心20中。核心20由GPU 12能够执行的原代码(例如，二进制指令)组成。编译程序/驱动程序18还可管理GPU 12的运行时执行。如下文更详细地描述，编译程序/驱动程序18可在根据本发明的技术的运行时处将发散栅栏指令插入到核心20中。CPU 16可将核心20发射到GPU 12以供进一步处理。

[0033] GPU 12可为允许大规模并行处理的专用硬件，所述硬件对于处理图形数据非常适

合。以此方式，CPU 16卸载由GPU 12更好地处置的图形处理。CPU 16可与符合特定应用处理接口(API)的GPU 12通信。此类API的实例包含Microsoft®的DirectX® API及科纳斯组织的OpenGL®；然而，本发明的方面不限于DirectX及OpenGL API，且可扩展到已开发、当前正在开发或待开发的其它类型的API。

[0034] 除定义GPU 12接收来自CPU 16的图形数据的方式之外，API可定义GPU 12将实施的特定图形处理管线。图1中的GPU 12说明Direct3D 11 API所定义的图形处理管线。如更详细地描述，图2说明OpenGL 4.x API的图形处理管线。

[0035] CPU 16及GPU 12的实例包含(但不限于)数字信号处理器(DSP)、通用微处理器、专用集成电路(ASIC)、现场可编程逻辑阵列(FPGA)或其它等效集成或离散逻辑电路。在一些实例中，GPU 12可为包含集成及/或离散逻辑电路的专用硬件，所述电路为GPU 12提供适合于图形处理的大规模平行处理能力。在一些情况下，GPU 12还可包含通用处理，及可被称为通用GPU(GPGPU)。本发明中所描述的技术还可适用于GPU 12为GPGPU的实例。

[0036] 系统存储器14可包括一或多个计算机可读存储媒体。系统存储器14的实例包含(但不限于)随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪存储器或可用以载运或存储呈指令及/或数据结构形式的所要程序代码及可由计算机或处理器存取的任何其它媒体。

[0037] 在一些方面中，系统存储器14可包含使CPU 16及/或GPU 12执行在本发明中归于CPU 16及GPU 12的功能的指令。因此，系统存储器14可为包括使一或多个处理器(例如，CPU 16和GPU 12)执行各种功能的指令的计算机可读存储媒体。

[0038] 在一些实例中，系统存储器14可被视为非暂时性储存媒体。术语“非暂时性”可指示存储媒体未体现于载波或传播信号中。然而，术语“非暂时性”不应解释为意指系统存储器14是不可移动的。作为一个实例，可从装置10移除系统存储器14，及将所述系统存储器移动到另一装置。作为另一实例，实质上类似于系统存储器14的系统存储器可插入到装置10中。在某些实例中，非暂时性存储媒体可存储可随时间而改变(例如，在RAM中)的数据。

[0039] CPU 16还可产生用于GPGPU应用程序的命令和数据，例如，用于光线跟踪应用程序的命令和场景数据、物理仿真或用于任何其它类型的GPGPU核心的数据。GPGPU应用程序(例如，核心20)还可使用图形API(例如DirectX或OpenGL)或使用更通用的计算API(例如开放计算语言(OpenCL)或OpenCompute或DirectCompute)来编译。CPU 16可将用于核心20的数据发射到命令缓冲器以供处理。在不同实例中，命令缓冲器可为系统存储器14的部分或GPU 12的部分。在一些实例中，CPU 16可经由专用总线(例如PCI-Express总线或另一通用串行或并行总线)将用于GPU 12的核心20的命令和数据发射到程序。

[0040] 为执行命令缓冲器中的核心20的所储存操作，GPU 12可实施图形处理管线。图形处理管线包含执行如由执行于GPU 12上的软件或固件定义的功能，及由经硬接线以执行极特定功能的固定功能单元执行功能。执行于GPU 12上的软件或固件可被称作着色器，例如着色器22。着色器22可在GPU 12的一或多个处理元件(也被称作“着色器核心”或“PE”上执行。因为用户可对着色器编程而以任何可想象方式执行所需任务(如同任何其它处理器)，所以着色器22向用户提供功能灵活性。然而，固定功能单元经针对固定功能单元执行任务的方式而硬接线。因此，固定功能单元可不提供大量功能灵活性。本发明的技术涉及执行GPU着色器22上的核心，例如核心20。

[0041] 一旦CPU 16将与渲染图形场景或执行核心相关联的数据和/或命令发射到命令缓冲器，GPU 12开始通过GPU 12的图形管线执行命令。GPU 12的调度器24创建执行与核心相关联的工作的基础单元的线程。调度器24将线程指派到着色器22的特定处理元件。调度器24还将线程分组到线程束中以供执行，且开始执行线程束。

[0042] 如上文所论述，如果由于执行流控制指令不同线程跳转到不同指令，那么线程束的线程发散。在发散线程束的情况下，调度器串行执行线程的每一集合。也就是说，GPU 12不再并行而是以群组串行地执行全部线程束线程，此损害GPU性能。

[0043] 为在线程束发散时改善GPU性能，编程器或编译程序/驱动程序18可将发散栅栏指令插入到核心20中。发散栅栏与GPU 12在运行时评估的处布尔表达式相关联。布尔表达式为评估为真或假的表达式。在不同实例中，布尔表达式可包含运算算术运算符，按位逻辑运算符和/或逻辑运算符。通过基于布尔表达式确定是否执行发散栅栏指令，布尔表达式在控制何时GPU应执行发散栅栏方面提供灵活性。布尔表达式评估为一种方式，其中发散栅栏指令不同于传统栅栏指令。也就是说，不同于执行传统发散栅栏指令(其中GPU在执行栅栏指令时始终停止执行线程束)，线程束在每一发散栅栏处并不必须停止，因为发散栅栏与布尔条件相关联，且发散栅栏通常位于控制流块中，所述控制流块还与布尔表达式相关联。用于发散栅栏指令的伪代码的实例为：

[0044] divergence_barrier(布尔表达式)；

[0045] 发散栅栏指令使GPU确定对于到达发散栅栏指令的线程束中的每一者中的至少一个线程，与发散栅栏指令相关联布尔表达式是否为真。如果条件对于至少一个线程为真，那么GPU 12暂停执行多个线程束中的每一者，基于活动线程的数目将线程束排序，且随后将非活动线程与活动线程调换以形成新活动/非活动线程束。GPU 12继续将非活动线程与活动线程调换，直到具有全部非活动线程的非活动线程束不可产生为止。一旦非活动线程束不可产生，GPU 12恢复执行线程束。如果GPU 12形成具有全部活动线程的线程束，那么GPU 12还可立即从队列释放且开始执行线程束。

[0046] 作为根据本发明的技术的一个实例，计算装置2的GPU 12可经配置以执行一种方法，所述方法包括：为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真；暂停执行具有对于其表达式为真的对应线程的每一线程束；以及确定对于其表达式为真的多个线程束中的每一者的活动线程的数目。方法可进一步包括：基于多个线程束中的每一者中的活动线程的数目将对于其表达式为真的多个线程束排序；将多个线程束中的的第一线程束的活动线程的线程数据与多个线程束中的第二线程束的非活动线程的线程数据调换；以及恢复执行对于其表达式为真的多个线程束中的至少一者。

[0047] 图2为说明根据本发明的技术的在多个处理元件上执行的线程束的框图。图2说明在多个处理元件42A至42N(PE 42)上执行的线程线程束40。PE 42可为一或多个着色器22(图1)的一部分。线程线程束(例如线程束40)可包括一线程群组，GPU调度器24可将所述线程指派到多个处理元件(例如，PE 42)以供执行。图2的每一PE可包括能够在特定时间对多个数据值执行单一指令(例如向量指令)的单指令多数据(SIMD)单元。PE 42还可支持对单一数据值执行单一指令，例如对单一浮点值的单一操作。

[0048] 线程束40还包含GPU 12的调度器指派PE 42以供执行的指令44。在一些实例中，指令44可存储于命令缓冲器中。指令44可包含每一PE经配置以经配置以执行的核心的指令

集。程序计数器(PC)50指示PE 42中的一或多者将执行的当前指令。在指令结束执行PE 42之后,PC 50的值可递增到核心20的下一个指令的地址。线程束40还包含寄存器46。寄存器46A至46N(寄存器46)可为能够处置多个数据值或单一值的通用寄存器。寄存器46可被“储备”,即,可加载和存储用于特定PE的数据。作为一实例,寄存器46A可限于存储用于PE 42A的数据,且可不加载或存储用于其它PE的数据。寄存器46中的每一者46可将数据供应到PE 42中的一者和/或供应来自其的数据,PE 42可随后处理数据。线程束40还可包含线程束上下文数据48。线程束上下文数据48可包含常见或在线程束40的不同线程当中共用的数据。作为一实例,上下文数据48可包含预测寄存器的数据,所述数据可包含用于执行于线程束40的PE 42上的每一线程的数据。

[0049] 线程束40、PE 42、指令44、寄存器46、上下文48以及PC 50可包括核心或GPU 12的着色器22的核心的部分。在不同实例中,线程束40可包括可为GPU 12的图形管线的部分的着色器(例如几何着色器、像素着色器和/或顶点着色器)的部分。在一些实例中,GPU 12可将由线程束产生的结果馈入到图形管线的另一阶段中以供额外处理。

[0050] 在执行线程束40上的核心期间,PE 42中的一或多者执行位于由PC 50指示的地址处的指令44中的一者。在执行指令期间,PE 42可从寄存器46读取一或多个数据值。PE 42可对数据值执行一或多个操作,且存储返回至寄存器46新值。PE 42可执行流控制指令,例如分支、跳转、转至等。流控制指令可使一个PE(例如,PE 42A)跳转到指令44中与PE 42B不同的一者,即执行于PE上的线程可由于流控制的不同评估而变得发散。然而因为存在单一PC 50,PE 42可在给定时间在一个特定处仅执行由PC 50指示的指令44中的一者。

[0051] 一旦线程束的线程发散,PE 42可仍仅在特定时间执行通过PC 50的值指示的一个指令。为支持发散执行,线程束40维持指示PE 42中的哪一者应执行PC 50的地址处的指令的状态,例如位掩码。作为一实例,PE 42A和42B可经调度以执行由采用“if-else”语句的不同分支产生的不同指令。在此实例中,PE 42A执行指令44中的第一指令,且PE 42B稍后执行第二指令,指令44中的第二不同指令。当PE 42A执行第一指令时,线程束40设定位掩码以指示PE 42A在执行指令期间为活动,而PE 42B为非活动。PE 42A随后继续执行指令44,直到PE 42A的线程结束执行或暂停执行发散栅栏指令且暂停执行线程为止。一旦PE 42A结束执行,线程束40改变位掩码以指示仅PE 42B为活动,将PC 50的值改变为PE 42B应执行的指令的地址,且随后PE 42B执行由PC 50指定的指令,直到线程暂停或结束执行为止。

[0052] 如上所述,本发明的技术包括发散栅栏指令,当所述指令执行时,可在多个线程束(例如线程束40)的线程发散时改善GPU 12的性能。发散栅栏指令可包括应用软件编程接口(API)的部分,例如DirectX 11API、OpenGL API、OpenCL和/或DirectCompute等。写入此类API中的程序可将发散栅栏函数的调用插入到使GPU 12执行发散栅栏指令的核心20中。

[0053] 编译程序/驱动程序18或操作系统还可将发散栅栏指令的调用插入到核心20的代码中。在不同实例中,用户可使用编译程序/驱动程序18编译核心20。在编译期间,编译程序/驱动程序18可分析核心20,且确定其中很可能发生发散的程序的位置和将显著影响性能的位置中的至少一者,且可在那些位置中的至少一者处插入发散栅栏指令。编译程序/驱动程序18可在很可能发生线程发散的位置和将显著影响性能的位置中的至少一者处在运行时(也被称作“绑定时间”)将发散栅栏指令插入到核心20的指令中。

[0054] 很可能发散的代码的一个实例可为光线跟踪应用程序的代码,所述代码包含于下

文中。在此实例中，发散栅栏指令(例如由编译程序或用户)插入以在执行后续光线跟踪伪代码时降低发散：

```
i = 0;
While ( i < dynamic_limit) { // dynamic_limit goes from 0 to 30
[0055]   divergence_barrier(i%10==0); //eliminate divergence after each 10 loops
   //traverse scene tree and do ray intersection calculation
}
```

[0056] 以上伪代码为GPU 12的多个线程和线程束可执行的循环的实例。每一线程可(例如)基于在光线跟踪场景中光线生成的反射的数目执行循环不同次数。因此，一些线程可在执行循环的几个迭代之后结束，而其它线程可继续执行环路最多环路的三十个迭代。

[0057] 在此实例中，GPU 12在每一循环迭代期间执行发散栅栏指令。发散栅栏指令包含GPU利用循环每一迭代评估的布尔表达式。如果对于线程束的至少一个线程布尔表达式评估为真，那么GPU 12仅执行与发散栅栏指令相关联的操作，例如线程束排序和线程压缩。在此实例中，布尔表达式i％10＝＝0在循环的每第十迭代期间评估为真。当对于线程束的一个线程布尔表达式为真时，GPU 12可调换来自不同线程束的线程以形成具有更多活动线程的新线程束，过程被称作“线程压缩”。

[0058] 每当与一个线程束线程的发散栅栏相关联的布尔表达式评估为真，GPU 12将与线程相关联的线程束(例如，线程束40)放入到队列或缓冲器中。一旦线程束被放入到队列中，GPU 12阻止线程束执行，且将队列中的线程束40排序。

[0059] 图3中极详细地说明基于每一线程束的活动线程的数目将线程束排序。GPU 12可基于每一线程束中的活动线程的数目，使用插入排序将线程束中的每一者排序。GPU 12将线程束排序使得具有更多活动线程的线程束经排序队列的前部，且具有较少活动线程的线程束经排序在队列的后部。

[0060] 当线程束添加到队列中或在不在栅栏处暂停的情况下完成之后，GPU 12随后对队列中的线程束执行线程压缩，即将来自具有较大数目的活动线程的线程束的非活动线程与具有较大数目的线程中的较小数目的线程束调换。GPU 12继续将来自具有较大数目的线程的线程束的线程与具有较小数目的活动线程的线程束调换，直到GPU 12无法产生“非活动”线程束为止。非活动线程束为具有全部非活动线程的线程束。当将非活动线程数据与活动线程调换时，GPU 12还可调换每线程上下文数据48(若存在)。一旦具有全部活动线程的“完全活动线程束”通过调换线程而产生，GPU 12将完全活动线程束从队列移除，且将其状态设定为活动及利用当前指令恢复执行完全活动线程束。在GPU 12结束线程压缩之后，包含部分活动线程束和完全非活动线程束的全部线程束被设定为就绪或活动状态。部分活动线程束还利用当前指令恢复。完全非活动线程可快进到当前控制流块的末端，其如果无指令遵循当前控制块，那么完全非活动线程束可立即结束执行。关于图4极详细地说明调换线程束当中的线程的过程。

[0061] 在一些实例中，为将活动线程与非活动线程调换，GPU 12可将非活动线程和活动线程所储存的寄存器数据存储在寄存器调换缓冲器52中。GPU 12随后将以前非活动线程的寄存器数据存储在以前活动线程的对应寄存器中。GPU 12还使用多路复用器54(“MUX 54”)

将以前活动线程的寄存器数据存储在以前非活动线程的对应寄存器中。更确切地说，对于与每一线程相关联的每一寄存器，多路复用器54(“MUX 54”)在所储存的非活动和活动线程的寄存器值之间多路复用，且存储返回至待调换的线程束的寄存器堆的值。在调换过程期间，DBS 50还可调换来自第一和第二线程束的每线程上下文数据48。在一些实例中，GPU 12可不利用寄存器调换缓冲器52来调换数据。而是，GPU 12可并行调换寄存器值而非将值存储在缓冲器中。

[0062] 在一些实例中，每一线程束可指与特定线程(被称作使用寄存器指针的“库”)相关联的一组寄存器46。GPU 12可存储指针的映射表。表的每一行或列可对应于特定线程束，且对应于线程束(视表布局而定)的行或列内的每一条目可存储将特定线程映射到寄存器46内的寄存器库的指针值。GPU 12可将指针的映射存储到用于上下文数据48中线程束的线程的寄存器库。在一些实例中，如果寄存器46由每线程寄存器库指针引用，那么GPU 12可通过简言调换两个线程的每线程寄存器库指针值，而非使用寄存器调换缓冲器52和mux 54调换两个线程的对应寄存器值中的每一者来调换每线程寄存器数据。

[0063] 在一些实例中，执行核心可频繁访问GPU 12的(例如)全局存储器和/或系统存储器14，或执行具有高量存取时间或时延的其它操作。在此情况下，包含发散栅栏操作的栅栏操作可暂停过于线程束以隐藏这些长时延操作，且执行性能可受损。为加速具有长时延操作的核心的执行，一旦活动线程束(活动线程束池)的数目达到某一阈值，GPU 12可立即执行线程压缩。

[0064] 一些核心可包含“传统”栅栏操作与发散栅栏操作的混合物。传统栅栏操作使到达栅栏的全部线程束暂停，且不同于发散栅栏不与GPU 12在运行时评估的布尔条件相关联。传统发散栅栏操作还不使GPU 12执行线程排序和线程压缩。对于包含传统栅栏和发散栅栏的混合物的核心，发散栅栏指令应产生传统栅栏操作。在具有传统和发散栅栏的混合物的核心中，GPU 12可在不等待线程束由于执行传统栅栏操作而暂停的情况下执行线程压缩。

[0065] 一些核心还可包含子例程调用。在子例程调用期间，GPU可将线程数据与具有与称为子例程相关联的不同调用堆叠的线程束调换。当发散栅栏操作包含在此类调用内时，子例程调用可存在问题。举例而言，第一线程束的线程可调用核心的第一线(例如，线10)处的子例程。第二线程束可调用核心的稍后执行点(例如，线20)处的同一子例程。子例程包含发散栅栏指令。

[0066] 由于执行介入指令和/或其它因素，当第一和第二线程束执行子例程内部的发散栅栏指令时，第一线程束和第二线程束的堆叠可彼此不同。在具有子例程内部的发散栅栏的问题的一个实例解决方案中，GPU 12可完全禁止具有子例程内部的发散栅栏。在另一实例解决方案中，GPU 12可实施逻辑以确保在执行子例程调用内部的发散栅栏指令时，执行具有发散栅栏指令的子例程的线程束具有相同堆叠。

[0067] 图3为说明根据本发明的技术的基于每一线程束内的活动线程的数目将线程束排序的概念图。图3的实例说明未排序的线程束80的数目。GPU 12响应于将与发散栅栏指令相关联的布尔表达式评估为等于真而将未排序的线程束80排序，如上文所描述。在图3的实例中，未排序的线程束80包含线程束82、84、86、88、90、92、94、96和98。在未排序的线程束80中，活动线程束以对角散列说明。非活动线程束线程在无任何散列的情况下说明。

[0068] GPU 12基于每一线程束中的活动线程的数目将未排序的线程束82排序。所得经排

序线程束在图3经说明为经排序线程束100。在未排序的线程束80中，线程束82具有最多活动线程(全部活动)，随后按顺序为线程束90、线程束88、线程束94、线程束84、线程束98、线程束92、线程束86和线程束96(全部非活动)。如图3中所说明，GPU 12使用插入排序将未排序的线程束80排序。基于每一线程束中的活动线程的数目的插入排序的结果在图3中经说明为经排序线程束100。在不同实例中，GPU 12可将未排序的线程束80存储在队列中，随后将队列中的线程束就地排序，此导致经排序线程束100为队列。在不同实例中，队列可实施为指针的链表。每一指针可指向特定线程束。为将链表排序，GPU 12可在链表中调换与线程束相关联的指针。

[0069] 图4为说明用于将来自一个线程束的活动线程与来自另一线程束的非活动线程调换的技术的概念图。在图4的实例中，GPU 12先前已将未排序的线程束80排序到经排序线程束100中。GPU 12将非活动线程与经排序线程束100的活动线程调换。GPU 12将非活动线程与活动线程调换，直到“非活动线程束”(即，具有全部非活动线程的线程束)不再可产生为止。将非活动线程与活动线程调换的过程被称作“线程压缩”。将经排序线程束100的非活动线程与活动线程调换的结果经说明为经压缩线程束120。

[0070] GPU 12基于两个线程束中的活动和非活动线程的数目将非活动线程与活动线程调换。在图4的实例中，GPU 12将来自具有更多活动线程的线程束的线程与来自具有较少活动线程的线程束的线程调换。在图4中，GPU 12将具有非活动线程的最左边的线程束的线程与具有活动线程的最右边的线程束调换。GPU 12继续从外向内调换来自不同线程束的线程，即，将来自具有更多活动线程的线程束的非活动线程与来自具有更多非活动线程的线程束的活动线程调换，直到非活动线程束不再可产生为止。调度器24恢复执行当时仍保持在队列中的任何和全部线程束。另外，每当队列的头端处的线程束含有全部活动线程时，调度器24释放位于队列的头端处的具有全部活动线程的线程束且开始执行线程束。

[0071] 通过将非活动线程与活动线程调换，本发明的技术形成具有大量活动线程的线程束以及具有全部非活动线程的线程束。具有较大数目的活动线程的线程束增大GPU 12的利用率和输送量。具有全部非活动线程的线程束还可增大GPU 12的输送量，因为非活动线程束可“快进”到当前控制流块的末端或在指令遵循当前控制块时结束执行。因此，在一些情况下，具有全部非活动线程的线程束可立即结束执行。因此，GPU 12可减少执行时间或停止执行此类非活动线程束，且利用与非活动线程束相关联的PE来执行调度器24确定可在那些PE上执行的不同线程束。

[0072] 图5为说明根据本发明的技术的用于处置多个发散栅栏指令的技术的概念图。因为发散栅栏通常位于还具有相关联的布尔条件的控制流块中，线程束可在GPU为线程束的任何线程将布尔条件评估为真时进入发散栅栏所在的控制流分支，或GPU可允许线程束穿过发散栅栏所在的控制流块且继续执行。如果线程束并不进入栅栏所在的控制流分支或如果对于线程束中的全部线程发散栅栏的布尔条件为假，那么线程束可穿过发散栅栏。在图5的实例中，GPU 12执行包含多个发散栅栏(被称作“DB1”“DB2”和“DB3”)的核心20。作为一个实例，核心属于光线跟踪应用程序。图5说明八个线程(线程140、142、144、146、148、150、152和154)通过核心的进度。线程140至154的条形图的长度指示每一线程是否已到达发散栅栏DB1至DB3中的一者，或已完全结束执行(“结束”)核心。

[0073] 在图5的实例中，DB1、DB2和DB3中的每一者位于核心中的不同点处。含有对应于

DB1、DB2和DB3的三个发散栅栏样本的一个实例伪代码包含于以下：

[0074]

```
i = 0;
while(i < dynamic_limit){ // dynamic_limit goes from 0 to 30
   divergence_barrier(i%10==0); //DB1 for each 10 loops
   i++;
   // do some work …
}
if (dynamic_codition) {
   divergence_barrier(true); // DB2 for long control flow block
   // do some heavy work
}
else {
   divergence_barrier(true); // DB3 for long flow control block
   // do some heavy work
}
```

[0075] 核心伪代码包含多个发散栅栏指令，DB1、DB2和DB3。发散栅栏指令中的每一者出现在分支语句或循环语句中。线程束执行核心20可依据它们进入的控制流块和布尔条件(例如，与发散栅栏指令相关联的栅栏条件)的评估而到达不同发散栅栏。线程执行核心20可在核心20的执行中首现遇到DB1，随后为DB2或DB3。

[0076] 相对于单一发散栅栏指令，当多个发散栅栏指令存在于核心中时，GPU 12可类似地处置线程束排序和执行线程压缩的过程。确切地说，调度器24可将到达同一发散栅栏的线程束一起分组到被称作“压缩池”的单元中。GPU 12可将线程束的线程压缩在压缩池中，所述线程束已到达同一发散栅栏指令。

[0077] 更确切地说，GPU 12使与线程束已到达的发散栅栏相关联的前缀与每一线程束相关联。作为一实例，到达第一发散栅栏的线程束可具有前缀“1”，到达第二发散栅栏的线程束可具有前缀“2”等。每一线程束还经指派指示线程束中的活动线程的数目的第二数目，例如后缀。作为一实例，如果线程束具有三个活动线程束，那么线程束经所指派后缀“三”(3)。

[0078] 前缀和后缀的组合形成GPU 12使用其来将线程束队列中的线程束排序的数目。作为一实例，用于GPU 12的线程束队列中可存在三个线程将排序。第一线程束已到达发散栅栏“2”且具有四(4)个活动线程。GPU 12可将数目“24”指派给第一线程束以用于排序目的。第二线程束可能已到达发散栅栏“1”且具有一(1)个活动线程。GPU 12可将值“11”指派给第二线程束。第三线程束可能已到达发散栅栏“1”且具有3(三)个活动线程。GPU 12可将13指派为用于线程束的排序值。GPU 12通过每一现场束的值将队列中的线程束排序。排序的结果可为使得第三线程束(具有排序值11)位于队列的头端处，第二线程束(具有排序值13)为队列中的第二个，且第一线程束(具有排序值24)位于队列的尾端处。因为具有排序值11和13的线程束具有相同前缀“1”，所有GPU 12可形成压缩群组。

[0079] 在GPU 12暂停全部线程束且将线程束插入在队列中之后(或在线程束不暂停在栅

栏上时结束执行),GPU 12通过将活动线程与非活动线程调换而对第一线程束群组执行线程压缩。换言之,GPU 12在消除关于后续发散栅栏的发散之前消除关于第一发散栅栏的全部发散,即GPU 12在移动到发散栅栏DB2、DB3等上之前消除关于发散栅栏DB1的发散。

[0080] 为消除与特定发散栅栏相关联的发散,GPU 12将第一线程束群组从队列分离,形成压缩池且对池中的线程束执行压缩。由于GPU 12执行压缩,因此GPU 12将线程束从压缩池释放且在执行时恢复线程束,使得GPU 12可在到达任何后续发散栅栏后暂停所释放的线程束。同时,含有其余线程束的队列继续接收任何发散栅栏上暂停的额外线程束。在循环情况下,GPU 12可在它们到达栅栏DB2、DB3或甚至再次到达DB1时暂停恢复的线程束。GPU 12将线程束添加到队列且就爱那个线程束与队列中的其它线程束排序,如上文所描述。

[0081] 当全部那些线程束暂停且插入在队列中时(即,队列再次变得完整),GPU 12对队列中的当前第一群组重复同一压缩过程,举例来说,此可适用于DB2。应注意,在GPU 12完成先前压缩且将全部线程束从先前压缩池释放之前,GPU 12可不具有队列中的全部线程束且开始另一轮压缩。因此,连续压缩过程之间不存在冲突。一旦全部线程束暂停于同一栅栏上,此形成队列中的仅一个群组,GPU 12可将暂停的线程束中的全部从队列分离且清空队列。

[0082] 因为暂停于位于队列的前部处的栅栏(例如,DB1)上的线程束稍后很可能撞击后续栅栏(例如,DB2/DB3),因此GPU 12可利用压实仅第一线程束群组的技术,以在为后续栅栏(例如,DB2、DB3等)执行压缩时能够将尽可能多的发散线程束一起分组到压缩池中。通过一次压缩一个栅栏,此技术可通过在后续发散栅栏的压缩期间实现压缩池中的较大数目的线程束的压缩而提高线程压缩的效率。

[0083] 在多个栅栏的情况下,GPU 12可较早开始对发散栅栏执行压缩,即,当队列不完整时,以同一方式且在上文所描述的同一条件下。那些条件可包含(例如)含有传统栅栏的核心程序,或引发频繁长时延操作。

[0084] 当多个发散栅栏存在于核心中且与线程束相关联的布尔表达式为至少一个线程束线程评估为真时,GPU 12将线程束放置到队列中且使前缀与线程束相关联。前缀指示线程束已到达的特定发散栅栏。作为一个实例,调度器24可将例如“1”的前缀附加到与线程束140、146、148、152和154中的每一者相关联的标识符,以指示那些线程束已到达发散栅栏DB1。调度器24可将类似前缀(例如,“2”、“3”)添加到线程束144和150,以分别指示那些线程束已到达发散栅栏DB3和DB2。

[0085] DBM 52将线程束140、142、144、146、148、150、152和154中的每一者存储在队列中。线程束140至154起初未排序且与基于线程束已到达的发散栅栏的前缀相关联。DBM 52起初基于与线程中的每一者相关联的前缀将线程束140、142、144、146、148、150、152和154排序,且基于前缀数字将线程束一起分组到压缩群组中。

[0086] 具有对应于最早发散栅栏(例如,DB1)的前缀的线程束的群组被称作“压缩池”。在图1的实例中,压缩池156包含线程束140、146、148、152和154,所述线程束中之全部已到达发散栅栏DB1,且因此包含同一前缀。

[0087] 如上文所描述,GPU 12基于前缀(其基于达到的发散栅栏数目导出)和后缀(其与每一线程束中的活动线程的数目相关)将压缩池156的线程束排序。在GPU 12暂停栅栏上的全部线程束(除了已结束执行的那些)并将暂停的线程束插入到队列且将队列中的线程束

排序中之后,GPU 12将第一线程束群组(其表示队列中的最前面的栅栏)从队列分离且形成具有此群组的压缩池。GPU 12随后通过将压缩池中的具有大量活动线程的线程束的非活动线程与压缩池中的具有大量非活动线程的线程束的活动线程调换来执行线程压缩,直到非活动线程束可不再从压缩池的线程束产生为止。一旦GPU 12结束任何新线程束的线程压缩,DBM 52将线程束从压缩池156释放以供执行。

[0088] 同时,队列可在它们恢复如上文所描述的执行之后继续接收暂停于任何栅栏上的线程束。GPU 12可将新近接收到的线程束排序且将其排序在队列中以及使用如上文所描述的插入排序将现有线程束。一旦全部线程束已暂停于发散栅栏上且移动到中队列,或结束执行且随后退出,GPU 12将当前第一线程束群组从队列分离以形成压缩池,且对压缩池执行压缩。

[0089] 一些核心应用程序可需要甚至执行核心的线程的调部。对步调(线程以所述步调执行)敏感的核心还可使发散栅栏的使用复杂化。举例来说,当执行发散栅栏已添加到其的此类步调敏感核心时,一些线程束可到达发散栅栏且暂停,而其它线程束可不暂停于发散栅栏处直到很久之后暂停在核心的指令序列中为止。因此,发散栅栏可导致不均匀的线程和线程束调步。为使具有第一相关联布尔条件的第一发散栅栏周围的线程调步均匀,编程器可插入具有为第一布尔条件的布尔补充的第二相关联布尔表达式的第二发散栅栏指令。

[0090] 以下伪代码说明此技术:

[0091]
```
 i = 0;
while(i < dynamic limit) {
 i++;
 //to eliminate divergence
 divergence barrier(diverg conditon==true);
 // do some work
 }
//warps do not hit barrier wait for those do
divergence barrier(diverg conditon==false);
// some heavy work.
```

[0092] 在以上伪码中,GPU 12执行包含与第一布尔条件相关联的第一发散栅栏指令的循环。代码包含环路外部的第二发散栅栏指令,所述环路具有为第一布尔条件的补充的第二布尔条件。因为第二布尔条件所述第一的补充,因此GPU 12将暂停第一或第二发散栅栏指令处的每一线程束,进而确保恒定的线程调步。

[0093] 图6为说明根据本发明的技术的用于执行发散栅栏指令的技术的流程图。GPU 12可经配置以执行图6中所说明的方法。在一些实例中,GPU 12可为多个线程束中的每一线程束确定对于每一线程束的对应线程布尔表达式是否为真(200)。GPU 12可暂停执行具有对于其表达式为真的对应线程的每一线程束(202),且确定对于其表达式为真的多个线程束中的每一者的活动线程的数目(204)。GPU 12可基于多个线程束中的每一者中的活动线程的数目将对于其表达式为真的多个线程束排序(206)。GPU 12可随后将多个线程束中的第

一线程束的活动线程的线程数据与多个线程束中的第二线程束的非活动线程的线程数据调换(208),且恢复执行对于其表达式为真的多个线程束中的至少一者(210)。

[0094] 在不同实例中,图6的方法可进一步包括在恢复执行多个线程束中的至少一者之前调换对于其表达式为真的多个线程的每线程上下文数据48。活动线程的线程数据可包括活动线程的寄存器数据,且非活动线程的线程数据可包括非活动线程的寄存器数据。在一些实例中,将多个线程束排序可包括使用插入排序将多个线程束排序。

[0095] 在一些实例中,GPU 12可为对于其表达式为真的多个线程束中的每一线程束进一步确定多个发散栅栏中的相关联发散栅栏,且基于每一线程束的相关联发散栅栏将多个线程束中的每一线程束分组到多个压缩池中。为将多个线程束排序,GPU 12可经进一步配置以将多个线程束排序,包括将属于多个压缩池中的同一者的多个线程束排序。在不同实例中,第一线程束和第二线程束包括属于多个压缩池中的同一者的线程束,且为恢复执行对于其条件为真的多个线程束中的至少一者,GPU 12可经配置以恢复执行同一个压缩池的至少一个线程束。

[0096] 在一些实例中,GPU 12可基于与多个线程束中的每一者相关联的发散栅栏进一步将前缀指派到多个线程束中的每一者,且为将多个线程束分组到至少一个压缩池中,GPU 12可基于所指派的前缀将多个线程束分组到至少一个压缩池中。

[0097] 在又其它实例中,GPU 12可进一步确定多个线程束包含具有全部活动线程的线程束;以及恢复执行具有全部活动线程的线程束。在又另一实例中,为将对于其表达式为真的多个线程束排序,GPU 12可经配置以将多个线程束存储在队列中,基于活动线程的数目将对于其表达式为真的多个线程束排序,且将经排序多个线程束存储在队列中。

[0098] 在又另一实例中,编译程序/驱动程序18可经进一步配置以确定很可能发生发散的位置和将显著影响执行在多个线程束上的核心20内的性能的位置中的至少一者。编译程序/驱动程序18可将发散栅栏指令插入到至少一个位置的核心中。图6的方法的布尔表达式可与在此实例中的发散栅栏指令相关联。

[0099] 本发明中所描述的技术可至少部分实施于硬件、软件、固件或其任何组合中。举例来说,所描述技术的各种方面可实施于一或多个处理器中,包含一或多个微处理器、数字信号处理器(DSP)、专用集成电路(ASIC)、现场可编程门阵列(FPGA),或任何其它等效集成或离散逻辑电路,以及此类组件的任何组合。术语“处理器”或“处理电路”可大体上指前述逻辑电路中的任一者(单独或结合其它逻辑电路)或例如执行处理的离散硬件等任何其它等效电路。

[0100] 此硬件、软件和固件可实施于相同装置内或单独装置内以支持本发明中所描述的各种操作和功能。另外,所描述单元、模块或组件中的任一者可一起或单独作为离散但可互操作逻辑装置而实施。将不同特征描述为模块或单元意图强调不同功能方面且未必暗示此等模块或单元必须由单独硬件或软件组件实现。确切地说,与一或多个模块或单元相关联的功能性可由单独硬件、固件和/或软件组件执行,或集成到共用或单独硬件或软件组件内。

[0101] 本发明中所描述的技术也可存储、体现或编码于计算机可读媒体(例如,存储指令的计算机可读存储媒体)中。嵌入或编码于计算机可读媒体中的指令可致使一或多个处理器执行本文中所描述的技术(例如,当由一或多个处理器执行指令时)。计算机可读存储媒

体可包含随机存取存储器(RAM)、只读存储器(ROM)、可编程只读存储(PROM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、闪存、硬盘、CD-ROM、软盘、卡盒、磁性媒体、光学媒体或其他有形计算机可读存储媒体。

[0102] 计算机可读媒体可包含计算机可读存储媒体,其对应于例如上文所列的有形存储媒体的有形存储媒体。计算机可读媒体也可包括通信媒体,其包含促进计算机程序从一个地点到另一地点的传送(例如,根据通信协议)的任何媒体。以此方式,短语“计算机可读媒体”大体上可对应于(1)非暂时性有形计算机可读存储媒体,和(2)例如暂时性信号或载波等非有形计算机可读通信媒体。

[0103] 已描述各种方面和实例。然而,可在不脱离以下权利要求书的范围的情况下对本发明的结构或技术作出修改。

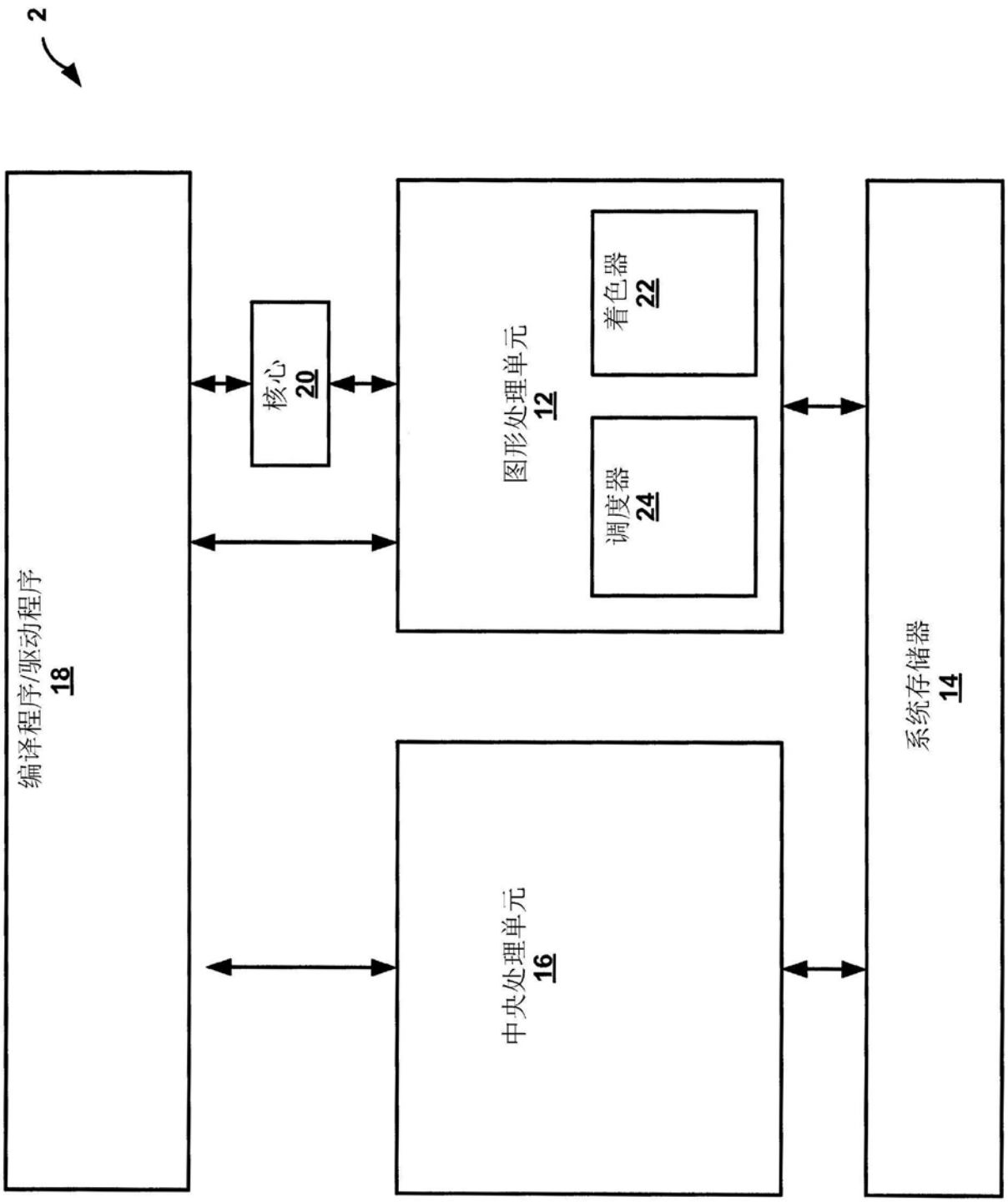


图1

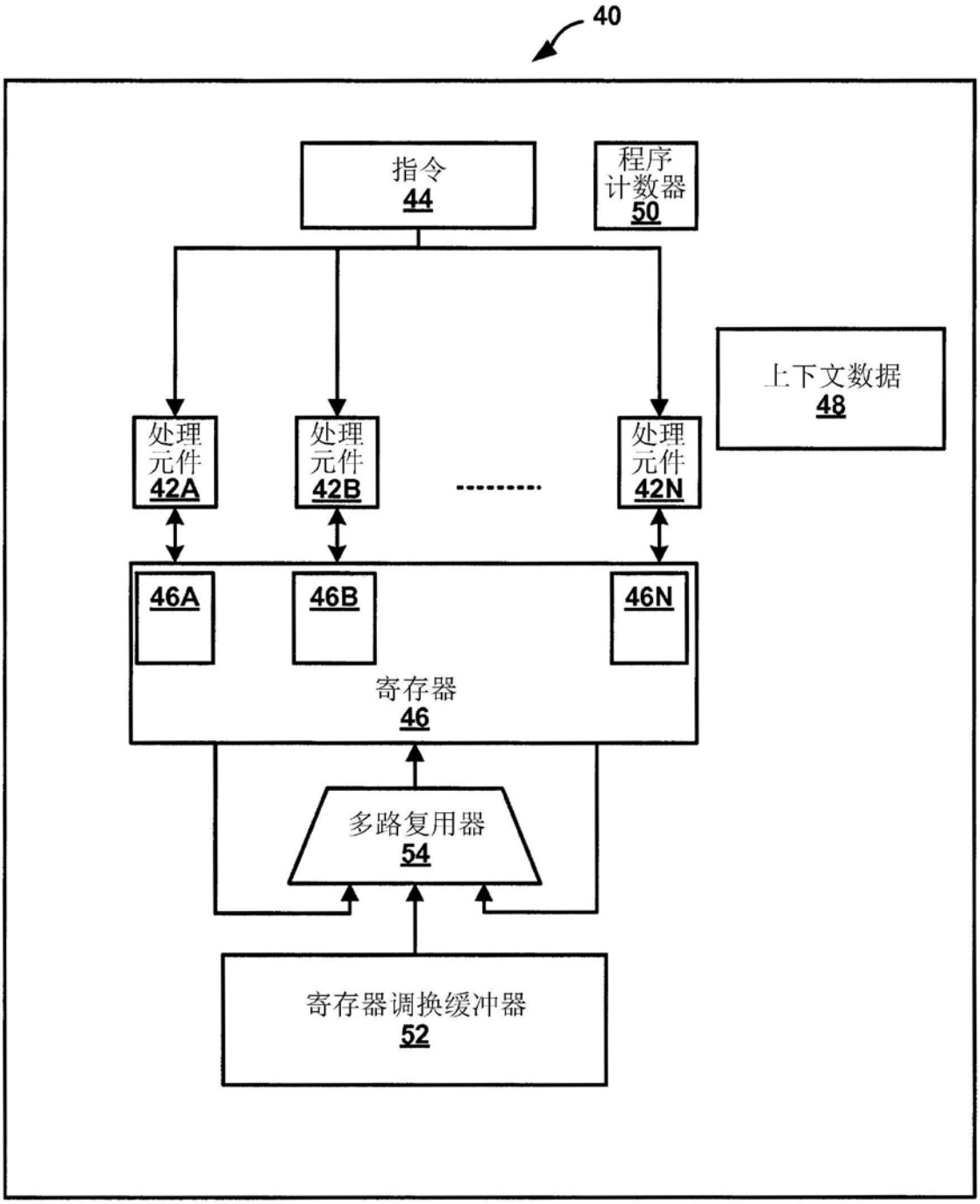

40
指令
44
程序
计数器
50
上下文数据
48
处理
元件
42A
处理
元件
42B
处理
元件
42N
46A
46B
46N
寄存器
46
多路复用器
54
寄存器调换缓冲器
52

图2

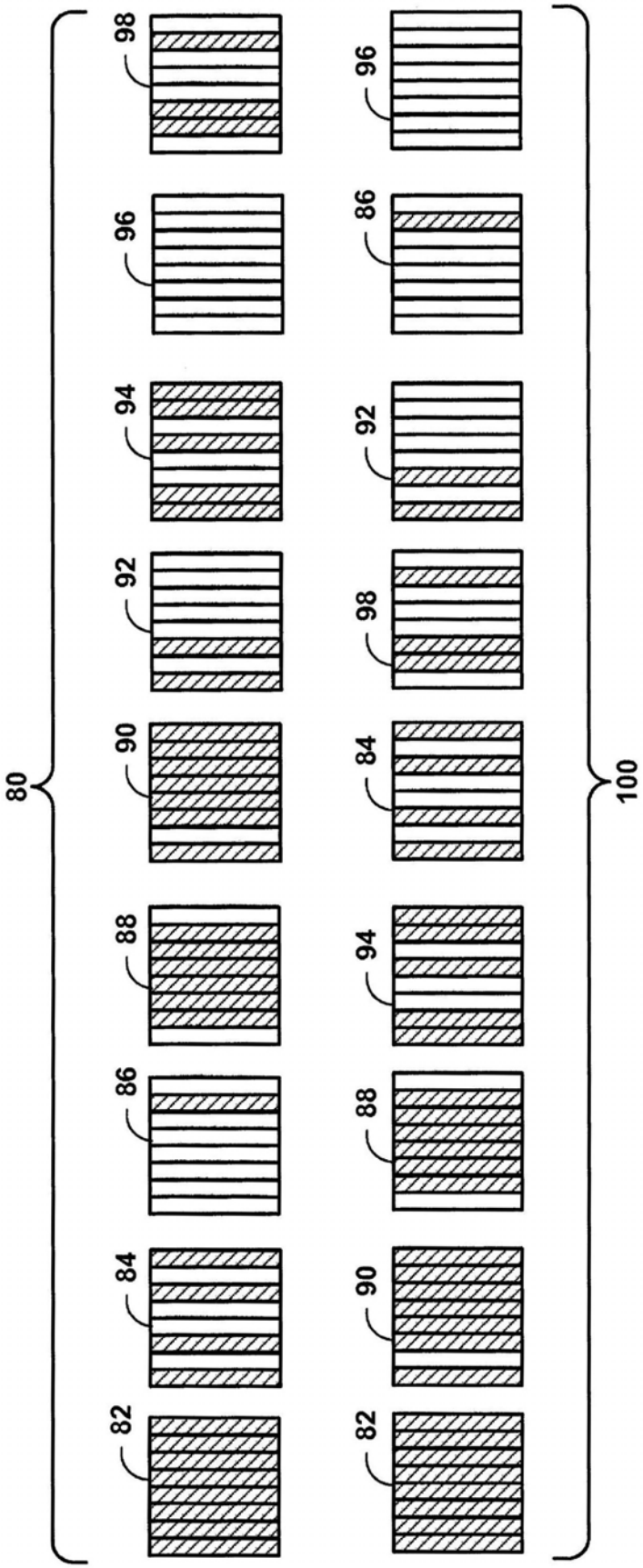


图3

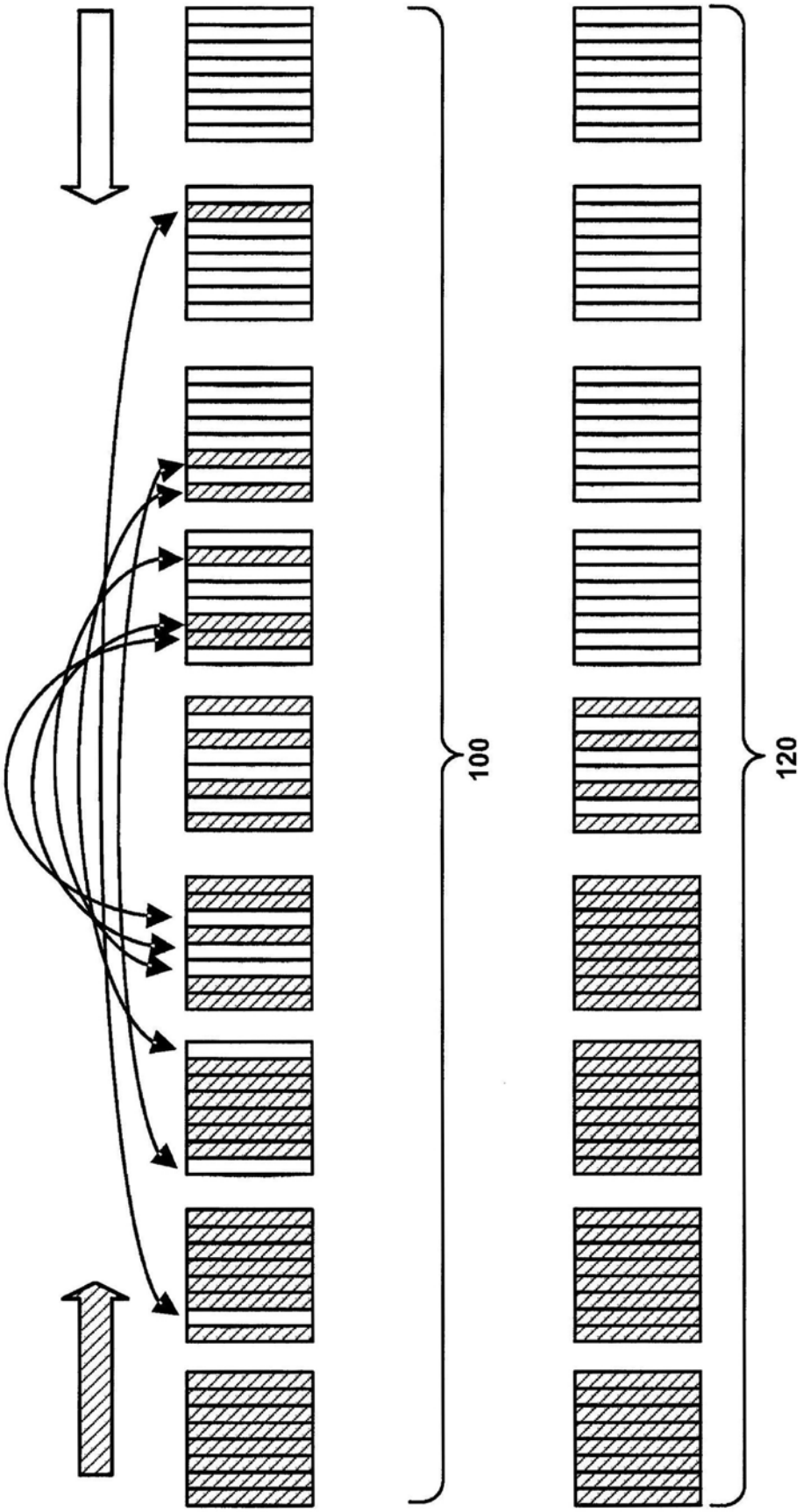


图4

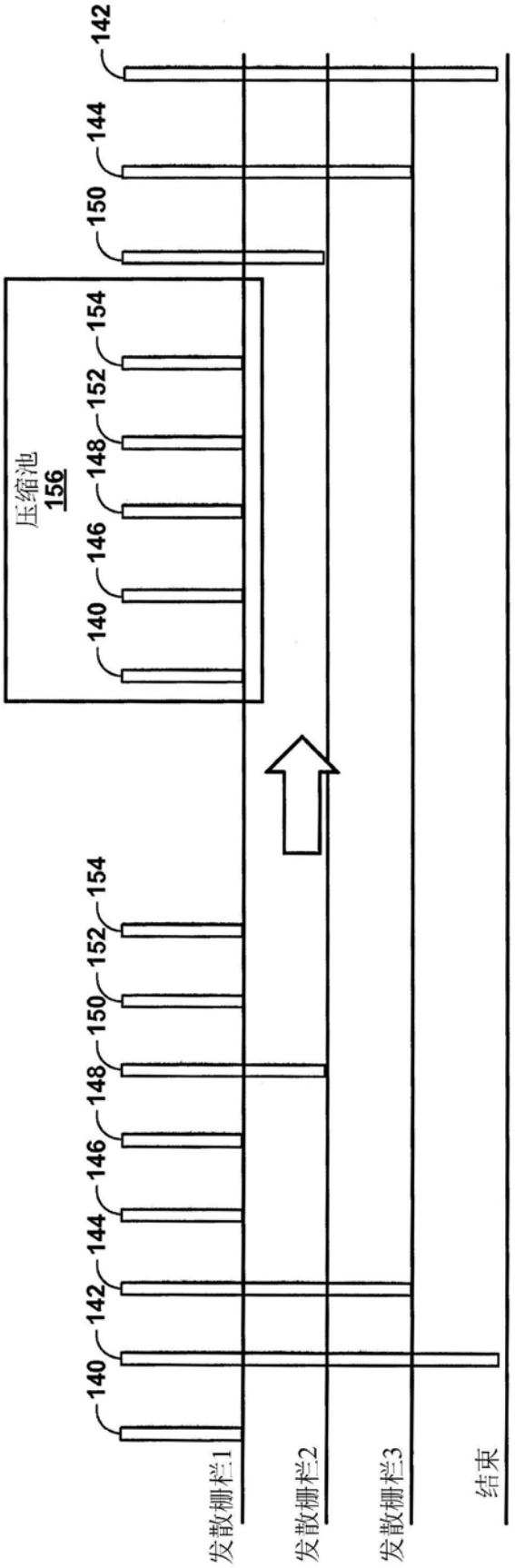

142
144
150
压缩池
156
154
152
148
146
140
154
152
150
148
146
144
142
140
发散栅栏1
发散栅栏2
发散栅栏3
结束

图5

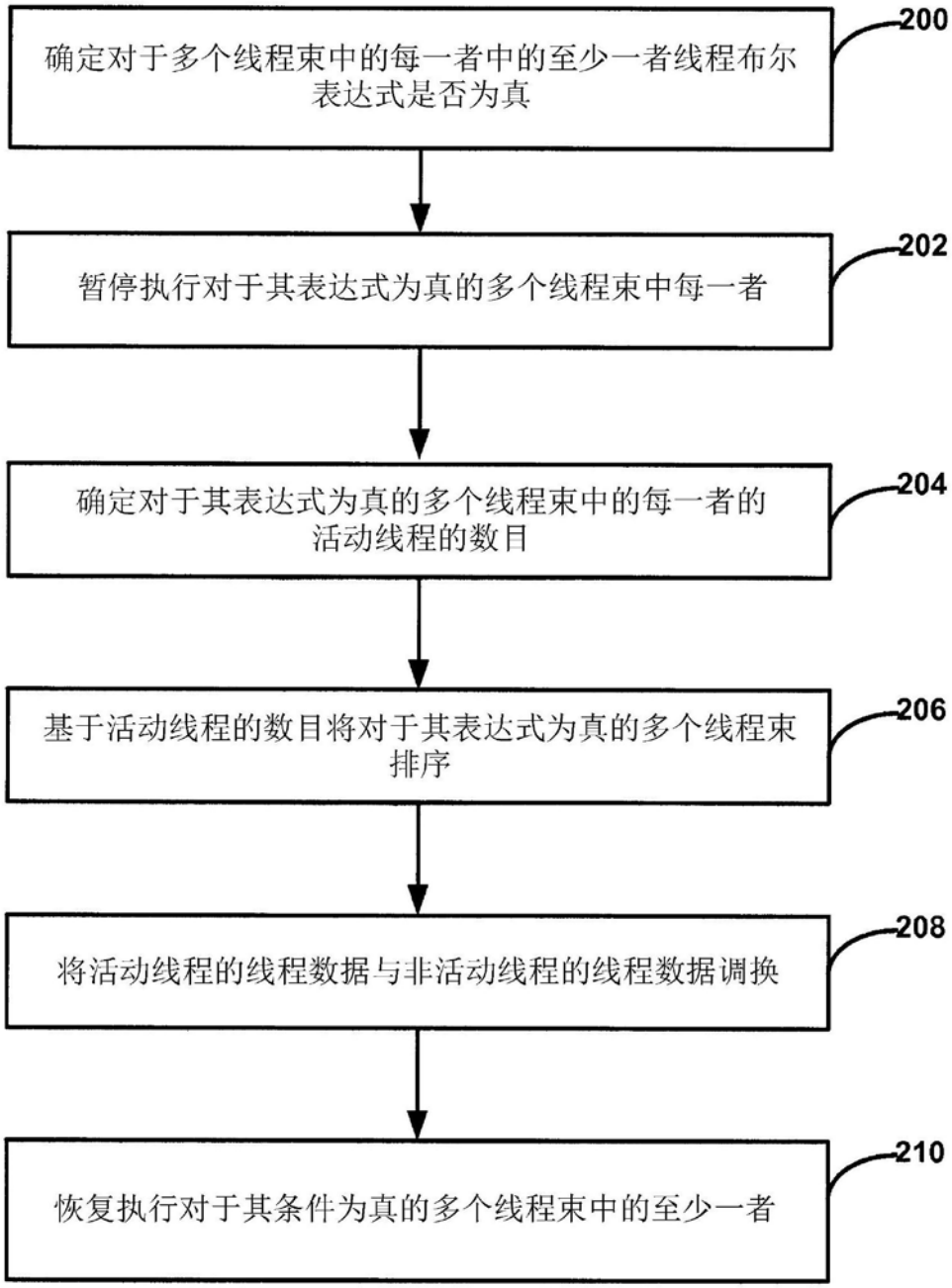

确定对于多个线程束中的每一者中的至少一者线程布尔表达式是否为真
200
暂停执行对于其表达式为真的多个线程束中每一者
202
确定对于其表达式为真的多个线程束中的每一者的活动线程的数目
204
基于活动线程的数目将对于其表达式为真的多个线程束排序
206
将活动线程的线程数据与非活动线程的线程数据调换
208
恢复执行对于其条件为真的多个线程束中的至少一者
210

图6